REMARKS

Applicant is in receipt of the Office Action mailed September 18, 2008. Claims 1-23 are pending in the case. Reconsideration of the present case is earnestly requested in light of the following remarks.

Section 103 Rejections

Claims 1 and 6-23 were rejected under 35 U.S.C. 103(a) as being unpatentable over Deutscher et al (US Pat. Pub. 2004/0001106, "Deutscher"), in view of Hampapuram et al. (US Pat. Pub. 2004/0221262, "Hampapuram").

Claim 1 recites:

1. A computer accessible memory medium which stores program instructions implementing a graphical user interface (GUI) for debugging a program, wherein, during execution of the program, the program instructions are executable by a processor to perform:

displaying source code for the program on a display during execution of the program, wherein the executing program was compiled from the source code;

receiving first user input hovering a mouse cursor over an expression in the source code during execution of the program;

in response to said hovering the mouse cursor over the expression, automatically displaying a GUI element proximate to the expression, wherein the GUI element includes a value of the expression;

receiving second user input to the GUI element modifying the displayed value, thereby specifying a new value for the expression; and

setting the expression in the program to the new value in response to the second user input, wherein the program continues execution in accordance with the new value of the expression.

Applicant presents below arguments made previously, as well as additional arguments directed to the Examiner's Response to Arguments.

Applicant respectfully submits that Deutscher fails to disclose, **displaying source code for the program on a display during execution of the program, wherein the executing program was compiled from the source code**, as recited in claim 1.

As explained in the previous two Responses, Deutscher is directed to a multimedia presentation production system in which presentation data are separated from presentation logic (see, e.g., paragraph [0008]). More specifically, per [0009], Deutscher discloses a tool which is

> ...essentially an intelligent visual data editor for making rich media presentations. Upon publishing from the tool, the required elements are organized and edited to contain the proper metadata in a single publishing directory. More particularly, **the tool consists of graphic user interfaces for easily entering the data associated with the rich media presentation. The tool is also designed to make it simple and easy to edit the data of the underlying schema**. (*emphasis added*)

As described in [0017]-[0018], among the various data entry graphical user interfaces (GUIs) disclosed are "data entry grids", which are displayed in a presentation tool window. As [0018] reads in pertinent part:

> ...**the data entry grids allow the user to enter the information into the presentation data file and the master track media file that is needed to drive the timeline of the presentation. The first of these grids**, which is displayed by default once the master track program has been imported, **is the scripts grid**. The scripts grid is where the user enters events called script commands that will be triggered during the playback of the presentation. (*emphasis added*)

The Office Action equates Deutscher's scripts grid with Applicant's claimed source code of an executing program, which Applicant respectfully submits is not correct, since, as Deutscher explains above, and as described in more detail in Deutscher's section 2.6.1 ([0137] – [0144]), the scripts grid is:

> **essentially a scheduling table** where the user enters events (sometimes referred to as script commands) that will be triggered during the playback of the presentation (*emphasis added*)

Note that the entered data in the various data grids (including the scripts grid) are saved to the presentation data file, which is then referenced by the presentation system to present the multimedia presentation. A scheduling table is *not* source code that is compilable to generate an executable program.

Applicant submits that this is a significant difference, since "setting the expression in the program to the new value in response to the second user input, wherein the program continues execution in accordance with the new value of the expression" is considerably more complex for compiled source code than for the grid in Deutscher.

In direct contrast to Deutscher's script grid, as one of skill in the programming arts readily understands, "source code" is a term of art in the programming domain and refers specifically to program instructions in a programming language that are compiled to produce executable code that may be run on a processor. This is emphasized in the independent claims by the limitation: "wherein the executing program was compiled from the source code".

Clearly, the cited script grid of Deutscher is not source code of a program, and more specifically, is not source code for an *executing* program, but, as described in paragraph 137, is a scheduling table, and more specifically, an editing tool for editing and providing data to a data file, which is then used by a presentation application to present a multimedia presentation. Nor does Deutscher describe any of the data grids as source code for a(n executing) program. Note, for example, that per paragraph 137, the program that is paused to allow for Deutscher's scripts grid editing is "a video or audio program that is designated as the master track", not an executable program for which the scripts grid is source code.

In the Response to Arguments, the Office Action asserts that Deutscher's script grid is equivalent to the source code of claim 1, arguing that "each perform [*sic*] the same functionality, Source code is read by the computer to perform the final outcome of the presentation. Script grid is read by the computer to perform the final outcome of the presentation. Each may be written, composed and used differently but essentially they perform the same function to produce a same end result, that they are both computer

instructions for performing desired entered task by the computer to output a desired result." This is not a proper analysis or characterization of the technical content of claim 1.

Applicant respectfully reminds the Examiner that *every word in the claim* is to be given consideration, and that the terms of the claim are to be interpreted in light of the Specification. More specifically, terms of art should not be re-interpreted contrary to the Specification.

Applicant submits that one of ordinary skill in the art would readily understand that a scheduling table (Deutscher's description of the script grid) used by an executing program to manage multi-media presentations does not "perform the same function" as source code that is compiled to generate the executable program. Note that substituting either of these elements for the other in their respective inventions would render the inventions inoperable, and so they clearly do *not* perform the same function, and are not equivalent. In other words, one of ordinary skill in the programming arts would readily understand that an editing tool and a data file used by an executing program (the presentation viewer) are *not* source code for the executable program.

For example, note that claim 1 particularly recites "displaying source code for the program on a display during execution of the program, **wherein the executing program was compiled from the source code**". As explained above at length, per Deutscher, the script grid is a data entry grid that allows the user to enter information (i.e., *data*) into the presentation data file. Now, the presentation data file is obviously a *data file*, which is used by the presentation viewer to present a presentation. In other words, the presentation viewer is the executable program that presents the specified presentation in accordance with the presentation data file, i.e., the presentation data file is or provides input data to the presentation viewer, which then operates accordingly. As one of ordinary skill in the programming arts would readily understand, the primary components or aspects involved in preparing and executing a program are: source code written in a programming language; an executable program compiled from the source code, i.e., the source code is compiled into executable program instructions; and (generally) input data, which informs the executable program as to specifically what to do. Now, Deutscher's presentation viewer is the executable program, presumably compiled from some source

code, which Deutscher does not discuss; the script grid is an editor that allows the user to generate an input data file, specifically the presentation data file, for use by the executable program–the presentation viewer. Note that the script grid produces data for the presentation data file, which is the input data for the executable presentation viewer. Note further that neither the script grid, nor the data produced by the script grid, nor the presentation data file, is compiled into executable code, i.e, executable program instructions. More specifically, the executable program–the presentation viewer–is *not* compiled from the script grid or the presentation data file, contrary to the Examiner's assertions.

The Examiner further argues that Deutscher shows that the user can input during execution of the program to edit the script bar, citing paragraphs [0137] and [0155], as well as Figures 13 and 17.

Paragraph [0137] describes the scripts grid–"essentially a scheduling table where the user enters events (sometimes referred to as script commands) that will be triggered during the playback of the presentation"–and clearly explains that this item is a means for providing input data for the presentation viewer (in the form of a presentation data file). Nowhere does this citation indicate that the scripts grid is source code that can be compiled into executable program instructions. Nor does this citation mention or even hint at the user editing these data during execution of the presentation viewer. Moreover, Applicant notes that even if these data were edited during execution of the presentation viewer, which Applicant argues they are not, this would still not teach editing the source code for an executable program, as explained above.

Figure 13 shows a text entry dialog box whereby the user can overwrite textual data for the script grid data table, but, as noted previously, the data do not include source code that is compilable to executable program instructions of the presentation viewer.

Paragraph [0155] relates to Figure 17, and is directed to editing transcription entries, wherein the user selects a data field, e.g., a time code field, and overwrites the data therein. Again, this citation in no way teaches that the transcription entries or the transcription grid in general comprise source code that is compiled to generate executable program instructions of the presentation viewer, nor that the transcription entries are edited during execution of the presentation viewer.

Figure 17 shows a text entry dialog box whereby the user can overwrite textual data for the transcription grid data table, including a transcription text field 1704.

The transcription grid is used to enter and edit a transcript of the audio portion of the master track for optional display during the presentation (from, paragraph [0151]). Moreover, as explained in paragraphs [0020] and [0021]:

> ...The transcription grid is where the user can enter a line-by-line transcription of the presentation in multiple languages for display in the final presentation. **Users can enter transcript information by entering the text into the grid with an associated start-time time code.** The user can choose how granular the timeline should be with the transcript grid. **Transcription can be spaced close together in short sentences or widely apart in large paragraph blocks.** A language dropdown box will also allow the user to select the associated language key with which this transcription data gets stored. The finalized presentation data file can store multiple language transcriptions, indexed by language keys.
>
> [0021] It is noted that in the foregoing scripts, markers and transcription grids, **the data can be entered by importing it via a formatted text-based file**. This method of **entering the data is especially useful with the transcription grid in that users can dramatically speed up the process of entering transcription information compared to manual entry methods. The file will be in a human readable format for easy integration with word processing, and spreadsheet applications**. Users import the file into the presentation production system and select the associated language key. **The data is then parsed into the presentation data file and is available for editing in the associated data grid.** (*emphasis added*)

In paragraph [0152], Deutscher states that the default language for the transcriptions is English, which is decidedly *not* a programming language. In other words, the transcription text is human-readable text, not program source code. Clearly, the contents of the grids are data, not source code that is compilable to executable program instructions.

While the transcription text field shown in Figure 17 includes what appears to be a for loop designation, note that this is actually text of a tutorial regarding programming in OLE and Visual Basic, as may be seen by examining the text of the transcription grid visible behind the data entry dialog box. Thus, the "for loop", which Applicant notes

includes generic "<code block>" designations, as well, is a textual passage that would be displayed to the user of the presentation viewer while the corresponding audio track of the tutorial is being presented audibly. Thus, the cited text is not intended for compilation, nor, Applicant notes, is it compilable at all, as one of skill in the programming arts can readily see.

Thus, the Examiner's assertion of equivalence between the source code of claim 1 and Deutscher's script (or transcription) grid is incorrect. Applicant further notes that the Examiner appears to assert this (incorrect) equivalence based on certain asserted high-level similarities between the two items, quoted above. However, this is not the proper way to analyze claim limitations. For example, note that if one considers any two items from a high enough level of abstraction, *everything* is equivalent. For example, a porkchop and an apple are both objects, and are both edible; this does not mean they are equivalent. Applicant respectfully submits that it is improper for the Examiner to disregard or redefine technical terms in Applicant's claims, thereby refusing to give patentable weight to these terms. Applicant respectfully requests that the Examiner give proper weight and meaning to the limitation "displaying source code for the program on a display during execution of the program, wherein the executing program was compiled from the source code", and submits that if such standard interpretation of these well-known technical terms is used, the Examiner's asserted equivalence will clearly be seen to be incorrect. Hampapuram also fails to teach or suggest this claimed feature.

Thus, for at least these reasons, Applicant submits that the cited art fails to disclose this feature of claim 1.

Nor does the cited art disclose **receiving first user input hovering a mouse cursor over an expression in the source code during execution of the program;** nor **in response to said hovering the mouse cursor over the expression, automatically displaying a GUI element proximate to the expression, wherein the GUI element includes a value of the expression**, as recited in claim 1.

As a careful reading of Hampapuram and Deutscher makes clear, neither reference discloses the claimed hover invoked functionality as claimed. As discussed above, Deutscher discloses editing a script grid, which, as Deutscher defines it, is

"essentially a schedule table", which is not source code as defined in claim 1. Nor is Deutscher's editing of the script grid performed during execution of the program. Applicant further notes that Deutscher teaches displaying the pop-up window in response to user input, specifically, double-clicking on the item to be edited, whereas in claim 1, the GUI element is automatically displayed in response to simply hovering the mouse cursor over the expression.

Nor does Hampapuram remedy these deficiencies of Deutscher. For example, per Hampapuram's Abstract, the macro expansions are processed by Hampapuram's tool during the build process of a project ("during a build of a programming project's source files"), where this processing operates to collect and record the macro expansion information into an output file or database. The tool then uses the recorded information to "display the macro expansions in a graphical user interface of the tool, such as for source browsing or viewing static analysis". Applicant notes that "source browsing" and "viewing static analysis" are not run-time operations, and are *nowhere described as being performed while the program is executing.*

Applicant respectfully notes that "expanding a macro" is not at all the same as automatically displaying a GUI element proximate to the expression (over which the user hovers a mouse cursor *during execution of the program*), where the GUI element includes a value of the expression. More particularly, Hampapuram's GUI does not display the value of an expression during execution of the program, but rather simply displays a macro expansion statically, i.e., *not* at runtime.

Thus, the cited art fails to teach or suggest these features of claim 1.

Nor does the cited art disclose **receiving second user input to the GUI element modifying the displayed value, thereby specifying a new value for the expression; and setting the expression in the program to the new value in response to the second user input, wherein the program continues execution in accordance with the new value of the expression**, as recited in claim 1.

As discussed above, neither Deutscher nor Hampapuram discloses displaying a value of an expression in source code of an executing program, i.e., at runtime. Nor do Deutscher and Hampapuram disclose modifying the value of such an expression in the

source code of an executing program, *where the program continues execution in accordance with the new value of the expression*. Note, for example, that Deutscher's editing of the script grid neither modifies an expression in source code as claimed, nor is performed during execution of the program. Similarly, Hampapuram's macro expansions do not teach or suggest modifying the value of an expression in the source code of an executing program, where the program continues execution in accordance with the new value of the expression, and is similarly not performed during execution of the program.

The Office Action asserts that it would have been obvious to one of ordinary skill in the art to combine Hampapuram and Deutscher because "one of ordinary skill in the art would recognize the program being used in the system of Deutscher does not have to be program specific for the functionality of a pop-up control and that the pop-up control could work in any program environment (e.g., a debugger)", and further asserts that "the combination of Hampapuram into Deutscher would yield the predictable result of having a control pop-up window which is initiated by hovering with the mouse courser over an area of interest by the user in such that the user is able to input data into the pop-up window upon presentation of pop-up window by the system [*sic*]".

Applicant respectfully disagrees.

The Examiner characterizes the functionality taught by a combination of Hampapuram and Deutscher as "having a control pop-up window which is initiated by hovering with the mouse courser over an area of interest by the user in such that the user is able to input data into the pop-up window upon presentation of pop-up window by the system". However, Applicant respectfully notes that there is a substantial difference between simply changing a value in a static editor as taught by Deutscher or expanding a macro in a source code browser or viewer as taught by Hampapuram, and dynamically modifying a value of an expression during execution of a program as claimed, i.e., via user input to a tooltip invoked via hovering the mouse over the expression during runtime, as claimed. For example, note that Deutscher edits text in the script grid, e.g, script type, script parameter, which is subsequently referenced by the program upon execution when the program accesses the presentation file to which the script grid's changes have been copied. This is not a dynamic process performed at runtime, and is

very different from Applicant's dynamic modification of a value of an expression during runtime, where the executing program continues to execute using the modified value.

Nor does Hampapuram's GUI facilitate editing the value of an expression during execution of the program, but rather simply displays a macro expansion statically, i.e., *not* at runtime.

Applicant respectfully submits that since neither reference discloses these features, one of skill in the art would not be compelled to combine them in an attempt to generate Applicant's invention, and so Applicant submits that a proper motivation to combine has not been provided, and thus, Hampapuram and Deutscher are not properly combinable to make a prima facie case of obviousness.

Moreover, even were Hampapuram and Deutscher properly combinable, which Applicant argues they are not, the resulting combination would still not produce Applicant's invention as claimed.

Thus, for at least the reasons provided above, Applicant submits that the cited art of Deutscher and Hampapuram, taken singly or in combination, fails to teach or suggest all the features and limitations of claim 1, and so claim 1, and those claims respectively dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable.

Independent claims 18-21 each includes similar limitations as claim 1, and so the above arguments apply with equal force to these claims. Thus, for at least the reasons discussed above, Applicant submits that claims 18-21, and those claims respectively dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable.

Applicant also asserts that numerous ones of the dependent claims recite further distinctions over the cited art. For example, nowhere does the cited art disclose **wherein the GUI element tooltip comprises: a first portion, operable to display the value of the expression, wherein the first portion is further operable to receive the second user input modifying the value; and a second portion, operable to display non-editable information related to the expression**, as recited in claim 10.

Deutscher's pop-up window only displays editable fields for editing the script information, and Hampapuram's pop-up window displays a macro expansion, and does

not allowing editing of the expansion. Thus, the references fail to disclose these features, and so claim 10, and those claims respectively dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable..

Applicant also asserts that numerous other ones of the dependent claims recite further distinctions over the cited art. However, since the independent claims have been shown to be patentably distinct, a further discussion of the dependent claims is not necessary at this time.

Removal of the section 103 rejection of claims 1 and 6-23 is earnestly requested.

CONCLUSION

Applicant submits the application is in condition for allowance, and an early notice to that effect is requested.

If any extensions of time (under 37 C.F.R. § 1.136) are necessary to prevent the above-referenced application(s) from becoming abandoned, Applicant(s) hereby petition for such extensions. The Commissioner is hereby authorized to charge any fees which may be required or credit any overpayment to Meyertons, Hood, Kivlin, Kowert & Goetzel P.C., Deposit Account No. 50-1505/5150-82801/JCH.

Also filed herewith are the following items:

☐ Request for Continued Examination

☐ Terminal Disclaimer

☐ Power of Attorney By Assignee and Revocation of Previous Powers

☐ Notice of Change of Address

☐ Other:

Respectfully submitted,

/Jeffrey C. Hood/
Jeffrey C. Hood, Reg. #35198
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert & Goetzel PC
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8800
Date: 2008-12-09 JCH/MSW